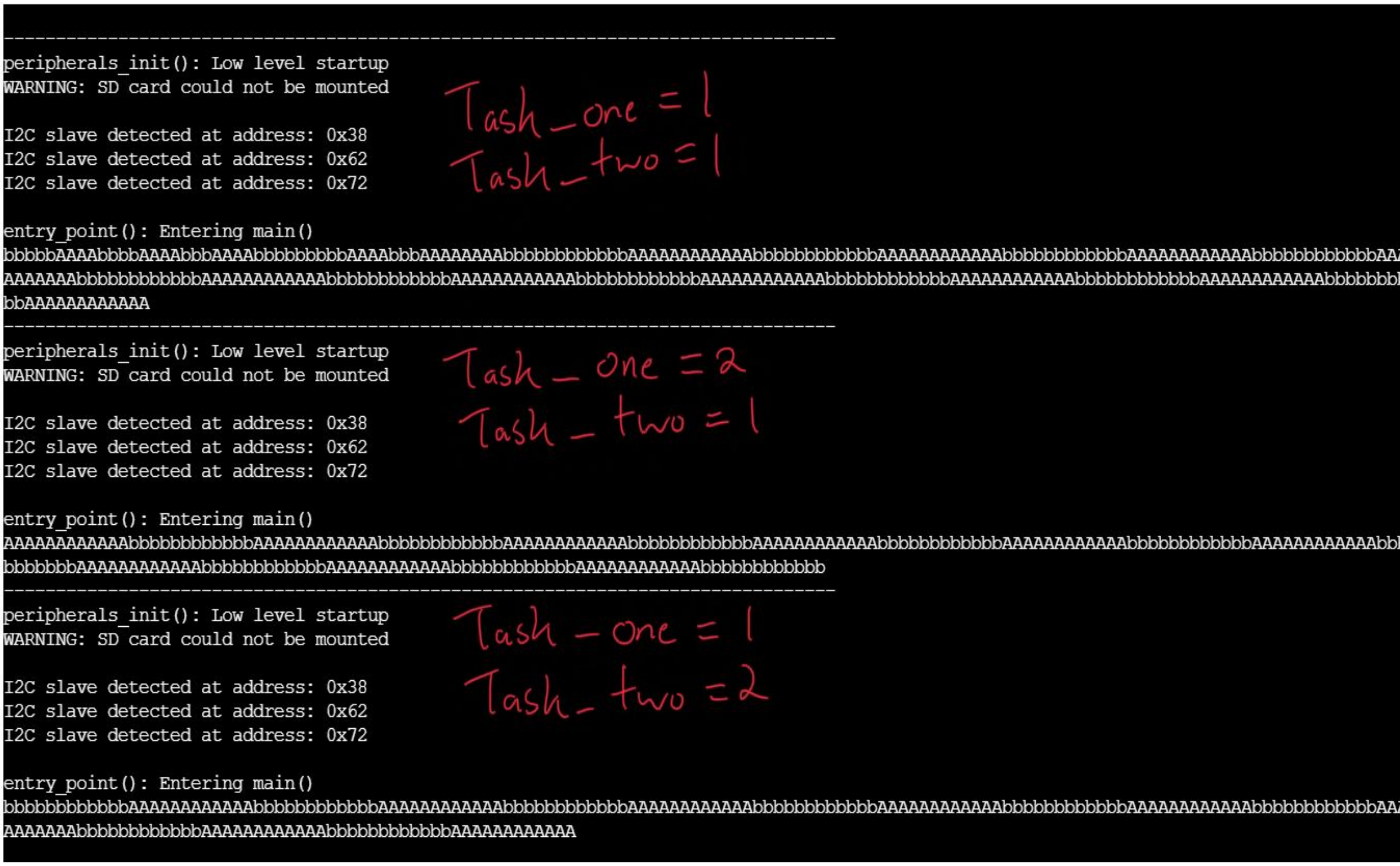
Jasper Tan
CMPE 146
2/8/2021
FreeRTOS Tasks Lab

Screenshots



Observation

* Same Priority: task_one = 1, task_two = 1

In this output task_one and task_two are both the same priority level. That means that task_one and task_two will get time sliced and follow a round robin schedule. This is very relevant to our tick rate of 1Khz meaning that we expect our CPU to alternate between task_one and task_two every 1ms (because 1Khz = 1ms). However keep in mind that at a baud rate of 38400 bps, that can send 3840 characters per second. Converting char per second to char per ms: (3840 char/1 second) * (1 second / 1000 ms) = **3.840 char per ms.**

This is the reason why 4 or sometimes 3 char are printed at any given time. It is because our tick rate is at 1Khz and our baud rate is 38400 bps. At these rates, we can only output 3-4 characters per tick (since 1 tick = 1ms).

* Different Priority: task_one = 2, task_two = 1
In this output task_one has a higher priority than task_two. Task_one outputs 'A' and task_two outputs 'b'. This priority level is reflected in our output because 'A' is printed 12 times first while 'b' is printed 12 times after. This cycle is consistent and continuous until ended.

* Different Priority: task_one = 1, task_two = 2
Now task_two has a higher priority than task_one. That means we expect to see 'b' printed before 'A' is. As you can see from the provided screenshots the assumptions are correct. 'b' from task_two is printed 12 times before 'A' from task_one is. This cycle is consistent and continuous until ended.

* Observation:
For the purpose of Lab: FreeRTOS Tasks create_blinky_tasks and create_uart_task are commented out
Prior to commenting them out, they somehow interfered and prevented our task_one and task_two from making expected
initial printouts. Ex. task_one = 1 task_two = 2 would print out 'bbAAAAAAAAAAAA' which is not expected.

Code
//From peripherals_init.c

```
uart__init(UART__0, clock__get_peripheral_clock_hz(), 38400);
```

\

//Relevant code to this lab

```
#include <stdio.h>

#include "FreeRTOS.h"
#include "task.h"

#include "board_io.h"
#include "common_macros.h"
#include "periodic_scheduler.h"
#include "sj2_cli.h"

// 'static' to make these functions 'private' to this file
static void create_blinky_tasks(void);
static void create_uart_task(void);
static void blink_task(void *params);
static void uart_task(void *params);

static void task_one(void *task_parameter);
static void task_two(void *task_parameter);
```

```
int main(void) {

  //For the purpose of Lab: FreeRTOS Tasks create_blinky_tasks and create_uart_task are
commented out
  //Prior to commenting them out, they somehow interfered and prevented our task_one and
task_two from making stable
  //initial printouts.  Ex. task_one = 1 task_two = 2 would print out 'bbAAAAAAAAAAAA' which
is not expected

  // create_blinky_tasks();
  // create_uart_task();

  // puts("Starting RTOS...Hello World Jasper here");

  xTaskCreate(task_one, "task_one", 1024, NULL, 1, NULL);
  xTaskCreate(task_two, "task_two", 1024, NULL, 2, NULL);

  // If you have the ESP32 wifi module soldered on the board, you can try uncommenting this
code
  // See esp32/README.md for more details
  // uart3_init();                                                                    //
Also include:  uart3_init.h
  // xTaskCreate(esp32_tcp_hello_world_task, "uart3", 1000, NULL, PRIORITY_LOW, NULL); //
Include esp32_task.h

  vTaskStartScheduler(); // This function never returns unless RTOS scheduler runs out of
memory and fails

  return 0;
}

static void task_one(void *task_parameter) {
  while (true) {
    // Read existing main.c regarding when we should use fprintf(stderr...) in place of
printf()
    // For this lab, we will use fprintf(stderr, ...)
    fprintf(stderr, "AAAAAAAAAAAA");

    // Sleep for 100ms
    vTaskDelay(100);
  }
}
```

```c
static void task_two(void *task_parameter) {
  while (true) {
    fprintf(stderr, "bbbbbbbbbbbb");
    vTaskDelay(100);
  }
}
```